APPLICATION OF STEM TECHNOLOGIES ON AVERAGE PROFESSIONAL EDUCATION

M.V. Rozhkova

Experience of complex studying of mathematics in technical college with use of modern information technologies is stated.

Keywords: mathematics, information technologies, computer algebra.

УДК 5530.12+531.51

ТЕХНОЛОГИЯ РЕШЕНИЯ ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ НА ОСНОВЕ СИНТАКСИЧЕСКОГО АНАЛИЗА ЕЕ ТЕКСТА

В.Э. Садриев[1]

1 *vsadriew@mail.ru*; Казанский (Приволжский) федеральный университет

Описана технология преобразования синтаксических конструкций естественного языка в соответствующие им конструкции на языке программирования.

Ключевые слова: программирование,естественный язык, задача, синтаксический анализ.

Как правило, текст подавляющего числа задач, в том числе и по программированию, строится с учётом всех правил грамматики и синтаксиса русского языка. Компьютерная программа - это тоже текст, но составленный по правилам некоторого языка программирования, который также имеет свою чёткую структуру, грамматику, правила. Таким образом, возникает вопрос, возможно ли найти соответствие грамматико-синтаксических конструкций естественного языка, каким является русский, соответствующие им конструкции в формальных языках, к которым относятся языки программирования.

При этом очевидно, что данные предположения могут быть справедливы для относительно несложных конструкций, потому что количество информации, заключенной в отдельные слова, сравнительно невелико, поэтому и соответствующие им конструкции в языках программирования также будут обладать сопоставимым относительно малым информационным объёмом.

Здесь предполагается провести сравнительный анализ языковых конструкций и проследить, каким именно образом будет изменяться синтез эквивалентных им участков кода программ. Здесь и далее под понятием «язык» будет подразумеваться текст на естественном языке, в первую очередь русском, а под понятием «код» будет предполагаться текст, созданный на одном из языков программирования.

Идея данного исследования возникла как результат преподавания курса программирования в школе в рамках уроков информатики. Основная проблема заключается в том, что часто учащиеся «не видят» в тексте задачи ни то, что дано, ни то, что нужно найти. Поэтому при разборе условия зачастую приходится акцентировать внимание на отдельных словах, словосочетаниях, а то и целых оборотах.

В стандартном предложении русского языка используются следующие основные грамматические конструкции: подлежащее, сказуемое, дополнение, определение, обстоятельство, причастный оборот, деепричастный оборот, остальные вводные конструкции. При составлении предложений обычно отталкиваются от подлежащего со сказуемым, к которым по определённым правилам добавляются остальные составляющие.

В общем случае код, записанный разными языками программирования, выполняющий одни и те же действия, внешне может быть записан с использованием различных технологий, кардинально между собой отличающимися. В рамках данной статьи будем оттал-

киваться от элементарных конструкций, используемых в языке Pascal и его современной разновидностью - Delphi.

Язык Pascal (паскаль) изначально был задуман для обучения программирования студентов благодаря чёткой структурированности и своей законченностью конструкций. Именно поэтому он был выбран для сопоставления.

К основным конструкциям языка программирования можно отнести: ввод-вывод, присваивание, ветвление, цикл, подпрограмму.

Для ввода данных в паскале используется команда *read*, где затем в скобках может быть указан список переменных для ввода. Знаковыми словами здесь являются «дано(ы) [число(а)]», «заданный [промежуток]». Вывод осуществляется командой *write*, где затем в скобках может быть указан через запятую список для вывода на экран или в файл.

Пример [1]: *Вывести на экран число π*.

Сразу стоит обратить внимание, что в постановке вопроса задач обычно отсутствует подлежащее, а глагол находится в повелительном наклонении или в форме инфинитива как в приведённом примере. Другие варианты постановки вопроса могли звучать как «Выведи на экран…», «Выведите на экран…». Последние варианты отличаются только тем, к кому конкретно обращается автор задачи: персонально к конкретному ученику или сразу к аудитории. Но это никак не влияет на то, как она будет решаться, поэтому будем использовать обезличенную форму инфинитива «Вывести на экран…». Применительно решением данной задачи будет одна команда:

```
Write(pi);
```

Таким образом, можно говорить, что сказуемому «вывести» соответствует команда *«write»*, где в скобках уточняется, что именно нужно вывести.

Перефразируем условие примера: «*Найти число π*». В данном случае получаем глагол «Найти…». Это ещё одна разновидность глагола «вывести».

Если подытожить, то первое, с чего нужно начать решение задачи учащимися - это с поиска сказуемого, которое указывает на то, что именно необходимо сделать, а также указывает на то, что нужно вывести на экран.

Что касается присваивания, то в чистом виде конкретных случаев наберётся всего лишь единицы, где необходимо вычислить конкретный пример. Более существенно будет присваивание рассмотреть в контексте отдельных составляющих математического выражения, которое можно сформировать из слов языка.

К самому распространённому существительному здесь можно отнести: число.

К самым распространённым прилагательным можно отнести: одно (первое), другое (второе), третье и т.д. [но тоже подразумевается число].

К самым распространённым глаголам можно отнести: удвоить, утроить и т.д., увеличить на…, уменьшить на…, увеличить в … раз, уменьшить в … раз.

При этом, если слова «первое», «второе», «оба» выступают в качестве подлежащего, то это будет означать, что в соответствующую им переменную в коде программы нужно именно присвоить, то, что дальше идёт уточняющим текстом.

Сведём эти слова в таблицу.

Слово	**Код**
число	x
первое	x
второе (другое)	y
третье	z
сумма	s или sum
количество	k или count
оба…, обоих…	begin x:=…; y:=… end
удвоить	*2
утроить	*3
увеличить на (прибавить) …	+…
уменьшить на (вычесть) …	-…
увеличить в … раз	*…
уменьшить в … раз	/…
кратно …	mod …=0
делится без остатка на …	mod …=0

Когда присутствует случай «если…, то…, иначе», то данный вариант очевиден и как правило соответствует тому, что даётся в стандартных учебниках. Случай «Если…, то…, иначе…» однозначно заменяется конструкцией *«if … then … else…;»*. При этом вариант «иначе…» - «else…» может отсутствовать.

Пример. *Если первое число больше второго, то первое удвоить, иначе из обоих вычесть 10.*

Решение. «Если» заменяется на «if», «первое [число]» - на «x», «второе» - на «y», «то» - на «then», «первое удвоить» - на «x:=x*2», «иначе» - на «else», «из обоих вычесть 10» - «begin x:=x-10; y:=y-10 end;».

Окончательно получаем:

```
if x>y
then x:=x*2
else begin x:=x-10; y:=y-10 end;
```

Ещё одно применение условия - это наличие вопроса в словесной конструкции «является ли…», «есть ли…», «верно ли». Соответственно, возможные ответы - либо «да», «есть», «является», «верно», либо «нет», «не является», «не верно». Для такого случая возможна следующая структура: *if … then writeln('да') else writeln('нет');*

Множественное число слова является признаком того, что отдельные действия происходят больше одного раза. Например, фраза «найти число» обозначает, что в результате решения задачи требуется вывести на экран одну переменную один раз, например *«writeln(x)»*. А вот фраза найти числа» подразумевает, что чисел явно больше, чем одно. На практике это обозначает наличие цикла. Таким образом, если постановка вопроса в единственном числе, то вывод результата производится уже по окончанию цикла, а если во множественном - внутри цикла.

Здесь будет рассматриваться цикл с предусловием *while*, так как он считается универсальным и любой цикл всегда можно свести к нему, при этом обратное не всегда справедливо. Далее уточним структуру цикла. С формальной точки зрения он внешне выглядит и поэтому преподносится учащимся [2], как структура из заголовка и тела цикла: *while условие do оператор;*

Данная структура не даёт глубинного понимания того, как проектировать его при прочтении задачи. Она удобна при анализе уже написанного цикла, но очень не достаточна при его синтезе. Мною же предлагается ввести другой способ преподнесения структуры цикла, которая даже затрагивает область за формальными его пределами.

Всё строится на понятии цикловой переменной, которую далее будем обозначать как i. Внешний вид цикловой структуры представим следующим образом:

```
Инициализация цикловой переменной;
While условие do
Begin
  Тело цикла;
  Коррекция цикловой переменной
End;
```

Видно, что вводится раздел инициализации цикловой переменной, а тело цикла разбивается собственно на тело цикла и на раздел коррекции цикловой переменной. В условии цикла как правило происходит сравнение переменных с какими-то значениями, но так как это не могут быть абстрактные значения, а всегда вполне конкретные значения, то значит, что до этого момента времени им нужно было где-то занести какое-то значение, то есть проинициализировать. Кроме того, чтобы не произошло зацикливания, внутри текущей итерации после завершения необходимых расчётов (тело цикла) требуется скорректировать значение цикловой переменной.

Инициализация, хотя формально и не является непосредственной частью цикловой структуры, играет важную роль. В ней задаётся стартовое значение для перебора в цикле в виде «цикловая переменная:=начальное значение». Например, для нахождения суммы трёхзначных чисел, стартовое значение можно задать как *i:=100*.

Условие между «while» и «do» обычно показывает конечное значение, заданного в задаче промежутка. Как правило, здесь происходит сравнение цикловой переменной именно с последним граничным значением в виде «цикловая переменная <= конечное значение». Для предыдущего примера о сумме трёхзначных чисел заголовок условия тогда примет вид *«while i<=999 do»*.

Тело цикла - это основная часть цикловой структуры. Здесь выполняется главная смысловая нагрузка задачи. Для её вычленения из условия задачи необходимо найти дополнение во множественном числе к сказуемому или другую конструкцию во множественном числе, которое является дополнением к дополнению (в единственном или множественном числе) к сказуемому. Основными типами задач, которые хорошо поддаются классификации - задачи на вывод набора чисел по условию, на нахождение суммы, произведения и количества.

В зависимости от типа задачи могут присутствовать дополнительные строчки кода в разделе инициализации, где, например, при поиске количества элементов по условию необходимо в переменную, отвечающую за количество, обнулить (проинициализировать). Помимо этого, после окончания цикла в некоторых случаях бывает необходимым что-то сделать с полученным результатом, например, вывести на экран.

Таким образом, это можно представить в виде таблицы

Естественный язык	**Дополнительный код в инициализации**	**Код в теле цикла**	**Код после окончания цикла**
Найти числа…	отсутствует	writeln(i);	отсутствует
Найти сумму…	s:=0;	s:=s+…;	writeln(s);
Найти произведение…	p:=1;	p:=p*…;	writeln(p);
Найти количество…	k:=0;	k:=k+1;	writeln(k);

Раздел коррекции также является необходимой частью цикловой конструкции. Он служит для изменения цикловой переменной для перехода к следующей итерации после завершения расчётов для текущего значения. Если происходит перебор чисел в восходящем порядке, то её значение увеличивается на 1, то есть *i:=i+1*, и уменьшается на 1 при переборе в нисходящем порядке, то есть *i:=i-1*.

Пример. *Найти сумму двузначных чисел.*

Решение. В условии дополнением к сказуемому стоит слово «сумму», к которому стоит

дополнение «чисел». Обозначим сумму через s, а числа через i. Поэтому в разделе инициализации будут команды обнуления суммы и задания начального значения цикловой переменной 10: *s:=0; i:=10*. В разделе условия нам нужно перебирать числа до максимального двузначного, значит *while i<=99 do*. В теле цикла находим сумму чисел *s:=s+i*. В разделе коррекции изменяем цикловую переменную *i:=i+1*. После цикла добавляем строчку для вывода на экран ответа *writeln(s)*. Итого окончательно имеем:

```
s:=0;
i:=10;
while i<=99 do
begin
  s:=s+i;
  i:=i+1;
end;
writeln(s)
```

При грамотном построении цикловая переменная должна встретиться не менее 3 раз, но обязательно в инициализации, в условии и в разделе коррекции цикловой переменной. Что касается приведенной структуры и внешнего вида цикла, то нужно сразу заметить, что в данном случае рассматриваются относительно простые виды циклов, используемые при обучении программированию в школе. В том числе понимается, что при некоторой степени допущения данную развёрнутую структуру можно свести к классической. Например, конструкцию бесконечного цикла *«while true do»*, где в условии явно отсутствует цикловая переменная, всегда можно записать в виде *«while i=i do»*, где уже цикловая переменная хоть и будет присутствовать, но сугубо формально.

Рамки данной статьи не позволяют подробно рассмотреть более сложные конструкции: массивы, записи, объекты, подпрограммы. Но из вышеприведенных фактов видно, что естественныя язык в принципе поддаётся описанию его через формат формального языка, коим является язык программирования. В данной статье был произведен анализ потенциальных возможностей. Его главной целью в является решение обратной задачи - задачи синтеза текста задачи по предлагаемой конструкции.

Литература

1. Златопольский Д. М. Сборник задач по программированию / Д. М. Златопольский. - 3-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2011. — 304 с.

2. Семакин И. Г. Информатика и информационно-коммуникационные технологии: базовый курс: учебник для 9 класса / И. Г. Семакин, Л. А. Залогова, С. В. Русаков, Л. В. Шестакова. - Москва: БИНОМ. Лаборатория знаний, 2005. - 371 с.

TECHNOLOGY SOLUTIONS FOR PROGRAMMING BASED PARSE ITS TEXT

V.E. Sadriev

The technology of converting natural language syntax in the corresponding construction for a programming language.

Keywords: programming, natural language, task, parse.